09/717,818.

WEST Search History

DATE: Friday, July 11, 2003

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=NO; OP=ADJ		
L25	l1 and l22	2	L25
L24	l22 and l4	9	L24
L23	database near5 migrator	1	L23
L22	custom$7 adj1 mapping	48	L22
L21	L20 and table$1	23	L21
L20	L19 and l1	35	L20
L19	client same server same migrat$5	245	L19
L18	L17 and (migrat$7).ab.	0	L18
L17	L16 and ((707/204)!.CCLS.)	18	L17
L16	table$1 same l6	2970	L16
L15	L14 and l1	4	L15
L14	l4 same l6	58	L14
L13	l6 same l2 same l4	0	L13
L12	l11	1	L12
L11	l9 and l6	1	L11
L10	L9 and l4	2	L10
L9	L8 and l7	16	L9
L8	l1 and l2 and l3	96	L8
L7	metadata	1253	L7
L6	reading same writing	54130	L6
L5	reading same writing same tables same metadata	1	L5
L4	migrat$4	103376	L4
L3	(second or target) adj1 database	1451	L3
L2	(first or source) adj1 database	1522	L2
L1	(707/204 OR 707/200 OR 707/100).CCLS.	2491	L1

END OF SEARCH HISTORY

WEST

End of Result Set

Generate Collection | Print

L5: Entry 1 of 1 File: USPT Apr 7, 1998

DOCUMENT-IDENTIFIER: US 5737549 A
TITLE: Method and apparatus for a parallel data storage and processing server

Detailed Description Text (36):
The invented storage server also stores compressed image files. The content of each image file extent is separately compressed, preferably at the client location. A compressed image file contains metadata which specifies the compressed size in bytes of each extent. The file system supports the storage of extents having a variable size. When a file is created in variable size extent mode, the DIB will contain information relative to the corresponding uncompressed file (image size, extent size, number of disks, disk table). In the same way as for uncompressed image files, FLEIB tables are generated at file creation time. In variable size extent mode, one set of continuous disk blocks is allocated dynamically when a compressed extent is to be written to a disk. The block address of the written extents is stored in the corresponding FLEIB table entries. Each FLEIB entry also contains a field specifying the effective compressed extent size in bytes. A library procedure exists for writing a compressed image file to a file previously created in variable extent mode. The parameters to be transferred by the server interface process are a table specifying for each compressed extent its size in bytes and a pointer to its memory location. The library procedure sends complete extent writing requests to the extent server processes. Extent server processes allocate the disk blocks required for writing their extents, write their extents on these blocks and update the corresponding FLEIB table entries. A similar library procedure exists for reading compressed image files from the disks. If an aligned window is requested by the reading procedure, only the complete extents covered by the aligned window are requested from extent server processes.

WEST

End of Result Set

Generate Collection | Print

L10: Entry 2 of 2 File: USPT Feb 18, 2003

DOCUMENT-IDENTIFIER: US 6523027 B1
TITLE: Interfacing servers in a Java based e-commerce architecture

Drawing Description Text (39):
FIG. 25 illustrates an environment migration process that guides development within ReTA engagement environments according to an embodiment of the present invention;

Drawing Description Text (87):
FIG. 73 illustrates that the code for technology architecture assembly test may be migrated from the technology architecture component test environment as defined in the migration procedures according to an embodiment of the present invention;

Drawing Description Text (97):
FIG. 82 illustrates the Migration Checklist Window according to an embodiment of the present invention;

Drawing Description Text (111):
FIG. 93.1 illustrates the PVCS Migration Flow according to an embodiment of the present invention;

Detailed Description Text (277):
FIG. 15 illustrates a method 1500 for handling events in a system. In operation 1502, an event which includes metadata is recognized. Next, in operation 1504, the metadata of the event is read and, in operation 1506 a table look-up is performed for information relating to the event based on the metadata. The information includes a severity of the event and further information such as a type of the event, and a location where the event occurred. In operation 1508, a message is displayed either in-line in a currently depicted display or in a separate display based on the severity of the event.

Detailed Description Text (492):
The project manager is responsible for the completion of the Project Configuration Management Plan during Design--with the help of the project team. This may: Clarify roles/responsibilities for migrations so that they are understood early in the project lifecycle. See FIG. 22, which illustrates the Configuration Management Life Cycle. First, a project study 2200 is created. Development and testing stages 2202,2204 follow the study. Finally, the implementation stage is reached 2206. Increase visibility of non-application components (e.g. database, architecture) in Configuration Management to improve quality of delivered products. Many times these are the components that are missed during implementations.

Detailed Description Text (497):
Change requests as a consequence of changing requirements and changes requested due to nonconformity (or defects), either in the application software, or in the system software must be analyzed, authorized, scheduled, staffed, and tracked in a defined way. What, why, when, and who made a change must be tracked from the point of analysis to the reintroduction of the defective or changed component at the appropriate stage. Change control therefore governs what software component is changed, version controlled, and when it is re-migrated to a given development stage.

Detailed Description Text (509):

During the second phase 2304, the Change Control Committee (CCC) meets regularly to review the change requests that have been logged to the Change Tracking tool in the past week. The committee also discusses the status of the changes scheduled for migration during the weekly migration windows, reviews the changes already moved to production, and sets the Staging Date for change requests. Before each weekly meeting, the Change Control Committee facilitator may generate the following reports: Report of the change requests that have been logged to the Change Tracking tool in the past week Implementation Report that list all changes scheduled to be implemented

Detailed Description Text (510):
During the meeting the CCC may: Review the new change requests Discuss the cross-functional impacts Verify that the target implementation date is realistic Set the Staging Date Update the status of the change requests scheduled to be implemented that week during one of the change windows Evaluate the quality metrics of the:changes that have been migrated to production and discuss any lessons learned

Detailed Description Text (515):
The Statement of Work/Scope Definition portion of the present description is sent to the change requester for sign-off. The sign-off needs to be checked-off on the Migration Checklist in the Change Tracking Tool in order to migrate the change to production. This sign-off serves as a quality checkpoint that the work on the change request may meet the business needs of the change requester.

Detailed Description Text (526):
The sign-off is needed to migrate the change to production. This sign-off serves as a final quality checkpoint that the work on the change request meets the business needs of the change requester.

Detailed Description Text (527):
Fill out Migration Form

Detailed Description Text (529):
In order to move the change to production, the developer needs to complete the Migration Checklist form on the Change Tracking Tool and inform Production Control 2316 by the Staging Date. This form contains all the information about the objects that need to be moved from the staging area into the production environment. This form is a streamlined checklist of all the things that the developers must do in order for Production Services personnel to move the objects to production. Whenever a sign-off checkbox is checked or unchecked, the current user's ID and the current date may be captured by the Change Tracking tool.

Detailed Description Text (530):
The following Migration Checklist items are required for the different change categories:

Detailed Description Text (531):
The Ready to Migrate checkbox is used to summarize that all the required sign-offs have been obtained and that the code is ready to be migrated to production. Finally, the developer should set the status of the change request in the Change Tracking tool to "Migrate".

Detailed Description Text (533):
Once Production Services personnel examines a completed Migration Checklist form, they may verify that all objects to be moved into production are in order, and that the change can be moved on the migration night in phase 2318. They may also ensure that all relevant items on the Migration Checklist have been completed. This check serves as the final quality checkpoint before the change goes into production.

Detailed Description Text (535):
Production Services personnel should set the status of each migrated change request in the Change Tracking tool to "Production". They should also set the Actual Implementation Date to the date the change was moved to production.

Detailed Description Text (537):
Business users and developers should continue to actively monitor the change requests

after it is migrated to production during phase 2320. If no problems develop in production due to the change request, the Change Control Committee may confirm that the team leader of the change request should set the status of the change request in the Change Tracking tool to "Closed". If problems do develop in production, the status should be set to "Re-Open". The developer is then re-assigned to fix the change request.

Detailed Description Text (539):
The Change Tracking tool contains metrics to track the quality of the change request. The Change Control Committee may assign the Migration Metric and Production Metric values for each change request approximately 35 days after it was migrated into production. If problems occur during the migration of the change request, the Change Control Committee may assign a "Fail" for the Migration Metric. The Problem Description should then be completed to explain why this problem occurred. The Lessons Learned should be filled with what lessons can be learned from the experience. If no problems occur, the Migration Metric may be assigned a "Pass".

Detailed Description Text (541):
Below are the criteria for the Change Control Committee to use in deciding if a change request passed or failed the migration metric or the production metric. A change request may pass if it meets the following criteria.

Detailed Description Text (542):
Migration Metric Criteria

Detailed Description Text (549):
Migration Control

Detailed Description Text (551):
Migration Control tools control multiple versions of source code, data, and other items as they are changed, tested, and moved from one development environment into another, for example, from development to test and from test to production. The list below provides a list of the various environments and their specific purpose within the project lifecycle.

Detailed Description Text (552):
With a ReTA/Microsoft-centric environment, a few key issues arise with respect to environment migration. These issues relate to the fact that the application is based on the use of Active Server Pages, Microsoft Transaction Server components and Java Classes.

Detailed Description Text (554):
To perform the code migration, certain steps should be followed to ensure that users that are currently in the application are not adversely affected. This can be accomplished by performing the migration in the following order: Using the Internet Information Server administration utility, monitor the site's number of active users. A count of zero indicates that no clients are currently hitting the site. Shut down the web listener to prevent additional users from connecting to the site.

Detailed Description Text (559):
There are basically three types of modules that get migrated during a ReTA engagement. Web Server files, Application files and database objects.

Detailed Description Text (561):
Application Server--Two file types are migrated within application servers, COM Dynamic Link Library's and Java Classes. Both files are created during the application and architecture build processes. The COM DLL's require registration within MTS by inserting them into a MTS.Package. In the event that the Web and Application servers are two physically different machines, an export process is required between them to instruct the Web server where the business components physically reside. For more information on the registration and exporting processes refer to the MTS online help.

Detailed Description Text (569):
Due to the security options available at both the Web and Application server levels, care should be taken during code migration to ensure that security settings are

consistent and applied correctly to ensure accurate execution.

Detailed Description Text (571):
Within MTS, every component has a transaction attribute that can be set by the MTS administrator to indicate what level of participation a component has within a transaction. Care must be taken during MTS component migrations to ensure that the correct transactional attributes are set within MTS.

Detailed Description Text (575):
During the ReTA Phase 1 engagement, Microsoft Visual SourceSafe was utilized for it's labeling and source code management capabilities. Additionally, the ReTA Change Tracker database could be utilized for source code migrations that required change management knowledge and approval. In the event that client requires the use of paper or email based migration control, the ReTA Migration Request template can be used.

Detailed Description Text (578):
The processes that guide development within ReTA engagement environments are represented in FIG. 25, which illustrates an environment migration process 2500. These processes include creating a new application 2502, modifying an existing application, and applying emergency bug fixes 2504. The solid lines represent stages required for new/modified application process. Dashed lines show the path for emergency bug fixes. Note: The term application used here is broadly applied to any managed module or component.

Detailed Description Text (582):
* -Stage is used to consolidate and verify vendor changes. Depending on the change, it may be migrated to Development or System Test 2506,2508 directly. The order may be dictated by project requirements.

Detailed Description Text (585):
The CM process depends on change control records (CCR) for tracking changes to the system. A change control record is created for every new module or modification. The CCR is used to coordinate migrations and communicate status for each module in the system. One may see the use of the CCR throughout every process description. The CCR processing system may be automated through Notes.

Detailed Description Text (608):
Version Control tools control access to source code as it is developed and tested and allow multiple versions to be created, maintained, or retrieved. For maintenance management purposes, it is desirable to designate one individual team member to function as the source control administrator. Duties for the source control manager would include the administration of source control users and projects, scheduling and performing periodic backups and applying labels to specific versions of the code (for migration purposes).

Detailed Description Text (615):
Additionally, this product provides: Easy to use drag-and-drop for file check in and check out Historical reporting and impact analysis User and project level security Archive and restore functionality Version `Labeling` for source code migration Support for web based applications

Detailed Description Text (771):
Care should be taken to allow for the migration from one vendor to another. To allow for this, the application developer should investigate encapsulating the component within an application wrapper.

Detailed Description Text (2165):
During the Assembly Test phase of a ReTA engagement, the Source Control Administrator may be responsible for the mass checkout and build of the entire application or architecture. FIG. 73 illustrates that the code for technology architecture assembly test may be migrated from the technology architecture component test environment as defined in the migration procedures. As shown, the test workstation 7300 is only connected to the web and application server 7302. The web and application server is connected to the source code repository 7304 and the database server 7306.

Detailed Description Text (2173):
As part of Assembly test, the following security roles may be created: Source Control administrator--responsible for monitoring code migration Web/Application server administrator--responsible for installation, configuration, maintenance and tuning on the server Database administrator--responsible for test database installation, maintenance and tuning

Detailed Description Text (2265):
The developer is in charge of configuration management (version control and migration control) of the components under their responsibility. When the component has successfully completed component test and code review, the developer should promote the code to the appropriate, staged location in the version control repository.

Detailed Description Text (2348):
As part of performance test, the following security roles may be created: Source Control administrator--responsible for monitoring code migration Web/Application server administrator--responsible for installation, configuration, maintenance and tuning on the server Database administrator--responsible for test database installation, maintenance and tuning

Detailed Description Text (2368):
Change Lifecycle A change request is submitted by a business user or IT personnel The Change Control Committee may review the change request. The change may be assigned to a developer and the status in the CTD may reflect the current status of the request. Once the change request is code/tested, it may be migrated into production. If it passes the monitor period its status may be closed. Otherwise, the change is re-opened and submitted through the process again.

Detailed Description Text (2387):
Migrating Window

Detailed Description Text (2388):
FIG. 82 illustrates the Migration Checklist Window 8200. This form allows the user to view general information 8202 about the change request and lists what sign-offs are needed in order for the change to be migrated to production. When a Migration Checklist item is checked or unchecked, the Sign-Off ID 8204 & Sign-Off Date 8206 may be filled in automatically with the current user's ID and the current date. Only the Capacity Planner should check the Capacity Planning Sign-Off checkbox.

Detailed Description Text (2527):
Legacy systems performance management tools are relatively well developed and provide a robust set of tools that manage all aspects of mainframe systems and communication networks. However, these existing legacy tools do not provide the same capabilities today for distributed networks supporting client/server and multimedia applications. It is important to understand the tool market before selecting an NPM tool. The state of the tool market can be summarized as follows: The client/server tools do not provide the mature and robust functionality of the legacy systems tools. Distributed systems are generally based on multiple vendor products and thus require management tools from a variety of vendors for full network performance management. Over the next few years, this situation may change as vendors cooperate and standardize through such associations as the Universal Measurement Architecture (UMA). The major legacy system management vendors are migrating their current products onto client/server platforms or developing new products to provide comprehensive tools that meet the different client/server and distributed environment needs. A number of different tools must be used to perform the full suite of network performance management functions in modern corporate networks. These include measuring, testing, monitoring, and simulating tools. Tools provide only limited and questionable information to network performance management personnel. It is difficult to classify tools into categories, as features and categories overlap.

Detailed Description Text (2559):
This portion of the present description details the use of PVCS for migration control within a project environment. The main objective of migration control is to manage the modules developed for a project. The migration process manages the development effort of multiple PVCS Users, by controlling the versions of source code as it moves from

development to production.

Detailed Description Text (2560):
The purpose of this portion of the present description is to satisfy the following criteria: Describe the migration control process for the development effort Define PVCS roles and responsibilities Portion of the present description the PVCS configurations for the UNIX and NT environments Explain the promotion model for the UNIX and NT environments Highlight the features and functionality of the PVCS migration control tool

Detailed Description Text (2561):
Detailed Migration Control Process

Detailed Description Text (2562):
Migration Control Environment

Detailed Description Text (2563):
The Development Architecture team designs the PVCS environment to manage the development effort. All modules under development (including database schema and object scripts; static HTML and images; Active Server Pages; JavaScript and style sheets; Interface Definition Language; Java source code; Rose Models; designs and supporting portion of the present description) should be version controlled and migrated using the PVCS migration process.

Detailed Description Text (2565):
Detailed Migration Control Process Flow

Detailed Description Text (2566):
FIG. 93.1 illustrates the PVCS Migration Flow, i.e., depicts the Migration Control process flow for the development environment. This diagram also shows a typical promotion model for the process. The three levels in the promotion model are development (DEV) 9330, test (TEST) 9332, and production (PROD) 9334. The following discussion is organized by promotion level.

Detailed Description Text (2572):
The PVCS Lead tracks modules ready for promotion. The PVCS Lead checks out and locks the modules that need to be migrated. A trial migration is performed to ensure that everything works as expected. Once this is complete, the modules are promoted from Development (DEV) to the Test (TEST) promotion level.

Detailed Description Text (2577):
If the tests are successful, the PVCS Lead is notified to promote the module to the PROD promotion level. The PVCS Lead checks out and locks the modules that need to be migrated. A trial migration is performed to ensure that everything works as expected. Once this is complete, the modules are promoted from the test directory to the production directory.

Detailed Description Text (2581):
The production promotion level (PROD) is the highest promotional level. This level contains modules that are thoroughly tested and ready to be moved into the production environment. When files are migrated to the PROD level, they are placed in the specified working directory a network server.

Detailed Description Text (2584):
Migration Control Process Roles and Responsibilities

Detailed Description Text (2585):
The Development Architecture team identified the three roles for the Migration Control process. These roles are PVCS User, PVCS Lead, and PVCS Administrator. This portion of the description defines each of the roles in relation to the Migration Control process.

Detailed Description Text (2592):
The PVCS Lead is a designated developer who coordinates the migration of modules from development to test and from test to production. The PVCS Lead works with each Cell

Lead (lead developer) to determine when the modules are ready to be promoted.

Detailed Description Text (2593):
Responsibilities Understand the working directories for the Build environment Communicate issues with the PVCS Users and PVCS Administrators to ensure that all problems are promptly addressed Ensure all modules are controlled by PVCS Portion of the present description all unsuccessful migration attempts Perform check in, check out, promote and demote functions Describe changes with version labels Ensure modules are compiled as expected

Detailed Description Text (2596):
The PVCS Administrator works with the PVCS Lead to ensure that the migration process works as designed. This person is responsible for the installation, configuration, maintenance, and troubleshooting of the PVCS application. The PVCS Administrator portion of the present descriptions the above activities.

Detailed Description Text (2597):
Responsibilities Train PVCS Users and PVCS Leads on the tool Communicate with the PVCS Users and PVCS Leads to ensure that all problems are promptly addressed Authorize, supervise, coordinate, and implement the actual migration design Test the configuration of the tool Work with the PVCS Leads to portion of the present description all unsuccessful migrations Portion of the present description all practices/lessons learned from the process Be aware of time schedules for critical times (e.g. server maintenance) Grant appropriate access to PVCS Users and PVCS Leads

Detailed Description Text (2598):
PVCS Migration Control Tool Description

Detailed Description Text (2600):
Intersolv's PVCS Version Manager can be used to implement the migration control process. This product may be referred to as PVCS throughout this portion of the present description. PVCS structures the development environment by providing the ability to access previous versions of the modules, create different releases of development code, and produce reports to track development effort. This portion of the description may highlight key features of the software and specify the software configuration for the UNIX and NT environments.

Detailed Description Text (2610):
Migration Control Procedures

Detailed Description Text (2679):
"Identify CM Units & Baselines" lists each component of the project that may be created, deleted, or otherwise modified. Along with identifying the configuration units, each unit type needs to have an associated promotion and migration procedure. At a minimum, the following types must be addressed on each project: design and test portion of the present description, database components, architecture components, and application components.

Detailed Description Text (2680):
A configuration unit is any object that is subject to reviews, deadlines, and/or utilized by multiple teams. These units should be classified by "type". For example, a set of batch programs could have 2 different "types": C programs and header files. Configuration types need to be defined in detail allowing changes to be planned, recorded, and verified. The CM plan should detail the review and migration process for each configuration type.

Detailed Description Text (2716):
The Technical Support team ensures that the project teams' development environment has been set up correctly, defines migration/promotion processes and resolves problems related to that, creates database environment for the project, and performs tests to ensure that the tools are functioning properly in the environment.

Detailed Description Text (2718):
The Test and Implementation teams are responsible for understanding the repository structure and migration processes defined by Tech Support. In addition, the

Implementation team is responsible for creating the release notice.

Detailed Description Text (2741):
Define Promotion and Migration Procedures 9604

Detailed Description Text (2745):
Migration--The physical movement of a kit/package from one environment to another

Detailed Description Text (2746):
Kit Build--The process of packaging the CM units so that they can be migrated to another environment.

Detailed Description Text (2750):
Standard Migration Paths

Detailed Description Text (2751):
The V-Model testing approach defines several testing environments. Based on this testing model, the program has defined the following standard migration paths. Component Test.fwdarw.Assembly Test.sup.* Assembly Test.fwdarw.Product Test.sup.* Product Test.fwdarw.Operational Readiness Test Product Test or ORT.fwdarw.Production.sup.* Production.fwdarw.Production Support

Detailed Description Text (2752):
Less complex systems, as well as extremely large systems, may not utilize all levels of testing. Projects should utilize at least the three base migration levels: Assembly Test, Product Test, and Production.

Detailed Description Text (2754):
When software products are moved from one environment to another it is important that the impacted parties receive sufficient notification. The vehicle used to deliver notification may vary across projects so it is necessary to portion of the present description in the Project CM plan how notification may occur. A release represents a move to another phase, such as a group of changes migrating to production. A release notice should be used for any modifications to configuration units or for the creation of new configuration units. The release notice should include a list of all identified problems and change requests that are being closed (i.e. changes being delivered) as part of this release; and should be created prior to migration to production.

Detailed Description Text (2755):
Critical Success Factors Appropriate repositories defined for size of project. CM units can be versioned and adequately controlled. Standard Control Process is used on the project. Access is controlled to appropriate repositories. Authorizer for creating a baseline is identified. The process for packaging, migrating, and installing is defined and portion of the present descriptioned.

Detailed Description Text (2757):
Repository Structure, Migration and Promotion Procedures, Packaging Procedures.

Detailed Description Text (2833):
CI Review may be scheduled on a regular basis as part of the Project CM Plan. The Program Manager may lead and facilitate the review meetings to assist the project team in gathering historical data to help assess the rate, causes and impact of changes. The content and format should be outlined in the Project CM Plan for the project. At a minimum, reports should be generated at the completion of each base migration level.

Detailed Description Text (2834):
Reports should contain the following types of information: Summary report of Change Requests by status and description. Specific Change Requests contained in each software version. Change history review of each configuration unit. A description of each configuration unit defined by its current release version. Change logs that show the history of releases and changes made to source files. Number of defects due to migration errors

Detailed Description Text (2842):
Measures Number of defects for project Number of defects for project due to migration

errors Number open, closed, deferred, rejected change request

Detailed Description Text (2896):
At a minimum, a Project CM Plan should contain information on the following: Definition of Configuration Units Types Baselines that may be established Unit unique naming standards Method for processing Change Requests Both System and Application repositories to be established and how they may be controlled CM related roles, responsibilities, and resources Definition of how objects are promoted/migrated between different environments Checkpoint meetings for project status and continuous improvement

Detailed Description Text (2923):
New Development/New Release Migration Process

Detailed Description Text (2924):
Program AT ->PT Promotion and Migration Guidelines

Detailed Description Text (3529):
FIG. 132 depicts a method 13200 for initializing a database used with an issue tracker. The issue tracker receives information relating to a plurality of issues from a plurality of users, displays the information relating to the issues, and allows the browsing of the information relating to each of the issues. To initialize the database, the information relating to the issues is stored in a first database in operation 13202. A second database is provided in operation 13204. The second database stores tables including a plurality of user interfaces and/or application logic for accessing the information in the first database. The tables of the second database are reconfigured in operation 13206 upon migrating the first database from a first folder to a second folder.

Detailed Description Text (3530):
As an option, a copy of the tables may be stored after being reconfigured. As another option, changing of a title of the first database may also be allowed upon migration from the first folder to the second folder. Additionally, the information relating to the issues may also be allowed to be edited.

Detailed Description Text (3534):
The Issue Tracker tool is comprised of two Microsoft Access databases. ReTA Issues DB--Client.mdb, which provides the user interfaces and supporting application logic and ReTA Issues DB.mdb, which contains the actual Issue Tracking data. To access the issue data the client database contains linked Access tables that actually reside in the second database. In order to function correctly these tables must be reconfigured so that the location references are correct. This step needs to be done every time the databases are moved to a difference file system folder.

Detailed Description Text (3648):
FIG. 157 illustrates an additional exemplary embodiment of a method 15700 for providing a resources e-commerce technical architecture. In operation 15702 issues in the technical architecture are managed for the purpose of resolution. A database used while managing the issues is initialized when migrated in operation 15704. Further, application consistency is maintained in operation 15706 by referencing text phrases through a short codes framework. In operation 15708, a plurality of software modules are generated in order to execute the technical architecture. Such software modules are based on business components.

Detailed Description Paragraph Table (70):
Title Description & Responsibilities Technical Typically an IS department head with responsibility for Manager the purchase and/or support of hardware and software. In configuration management, this role is more software oriented. Other responsibilities include: Assign development and support staff to projects. Review (accept/reject) technical approach proposed for projects. Monitor development and support budgets and personnel - status of projects. Network This individual is responsible for the installation, main- System tenance and support of the Unix and Windows NT Administrator servers including operating system, file systems, and applications. Other responsibilities include: Operating system installation, patch updates, migrations and compatibility with other applications. Installation and support of

proper backup/restore systems. Installation and support of other peripherals required for installed (or to be installed) applications. Proper portion of the present description of hardware configuration and setup. Maintenance of Windows Domain users and Groups as well as other security issues. Database The DBA is responsible for proper creation and main- Administrator tenance of production and system test databases. The integrity of the database, as well as recovery using backup/restore and logging, are priorities for the DBA. Other responsibilities include: Assist developers in maintaining development databases by automating backup/recovery, applying changes to database schema, etc. Provide support for tuning, sizing and locating database objects within allocated database space. Applying change requests to databases. Ideally maintain entity relationship diagrams for databases. Maintenance of database users and other database- related security issues Source Code Individual responsible for development and maintenance Librarian of source code control tools, training materials, and storage areas. The Source Code Librarian is also responsible for the integrity of the source code environment. Additionally: Establishes source code directories for new projects. Provides reports on source code environment status and usage per project. Provides assistance/information as needed regarding objects to check out for system test. Assists production operations in building/moving all applications into production. Business Individual or individuals responsible for managing the Analyst detailed design, programming, and unit testing of appli- cation software. Other responsibilities include: Developing/reviewing detailed designs. Developing/reviewing unit test plans, data, scripts, and output. Managing application developers. Application Individual or individuals responsible for making changes Developer to source code defined by management. This person typically: Checks source code out of the source code environment. Modifies code per user requirements or other develop- ment portion of the present description. Unit tests modifications in the development environment. Checks modified code back into source code environ- ment in preparation for system test. System Tester This person or team is directly responsible for system Integration testing or integration testing of an application prior to Tester implementing in production. This may also take the form of performance testing. Typically, a system or integration test person or team may be responsible for: Following production operation procedures for installing a new application in the appropriate test environment. Develop and execute a test plan to properly exercise new application including new, modified, and unmodified functionality. Reporting results of test. Vendor For the purposes of this portion of the present description, a vendor is defined as an organization from which software has been purchased for use by the clients systems. Alternatively, a vendor may distribute final installable media in the form of tape or CD with upgrades or new release of application. A vendor may: Make modifications to application code at vendor offices or within the engagement development environment. Provide necessary information to Source Code Librarian to store new code. Assist Source Code Librarian in transferring modifi- cations to the engagement system test environment. Participate in system test (or performance test).

Detailed Description Paragraph Table (72):
Checklist Item Project Enhancement Emergency Statement of Work Required Not Required Not Required Scope Definition Not Required Not Required Required User Acceptance Test Required Required Not Required Tech/Code Review Required Required Not Required Complete Portion of the Required Required Not Required present description Complete Components Required Required Required Submit Production Move Required Required Required Distribution Lists Required Required Not Required Requirements (TCPIP, Special Forms, Microfiche, Electronic Files) Identify Impacted Systems Required Required Not Required Capacity Planning Required Required Not Required Ready to Migrate Required Required Required

Detailed Description Paragraph Table (128):
Field Name Field Description CR# Automatically assigned when a new change request is entered. Date Date the change request was entered. This date defaults to the date of the change request entry. Logged By Portion of the present descriptions who entered the change request into the CTT Requester The person who requested the change request. Phone Phone number of the requester. Number Business Business area of the requester. Area Platform The hardware platform of the system affected by the change request. Appl. The application affected by the change request. Function The function affected by the change request. Component The component affected by the change request. Source Source of the problem: ABEND (Abnormal Program Termination) Other Performance Prior Change Regulatory Prior CR# Change request number of a previous request that caused

this current request or is related to the current request. Status The status of the change request. A change request can have a status of: New Assigned Development Testing Migrate Production Closed Rejected Deferred Re-Opened Priority The priority of a change request: 1 (High) - Change request is necessary for application functionality and is an integral component that keeps the system running properly. 2 (Medium) - Change request is severely needed for proper application functionality. 3 (Low) - Change request can be circumvented but needs to be resolved in the near future. 4 (Cosmetic) - Change Request does not affect production but should be fixed. Completion Requester's estimated date for the change Date request to be completed. Risk The risk of the change request: High, Medium, Low Impact The impact of the change request: High, Medium, Low Complexity The complexity of the change request: High, Medium, Low IT Area* Project or Area assigned to complete the change request: Account Management Client Services Delivery Systems Insurance/Corporate Systems Network Services Production Services Technical Services Category* The classification of the change request: Project A major change to the production environment, including application code, system software, hardware, and networks. Generally requires more than 160 hours of work. Generally tend to have high impact, risk, and complexity. Enhancement A minor change to the production environment, including application code, system software, hardware, and networks. Generally requires less than 160 hours of work. Generally tend to have low impact, risk, and complexity Emergency The application is out of service and there is no work around A security system can be or has been comprised Data loss/corruption Hardware failure that needs to be replaced immediately Site The site of the change request. Manager Manager responsible for change request. Assigned 1 Primary person assigned to complete the change request. Assigned 2 Second person assigned to complete the change request. Short A short (75 chars max) and concise description Description of the change request. Long.A detailed description of the change request. Description Target Date Date by which change should be ready to be migrated. Staging Date Date by which change should be ready to be migrated. Actual Date Actual date change is moved into production. Actual Hours Actual number of hours it took to complete the change request. Resolution The resolution to the change request. Developers should include a brief description of the changes made to the code. Explanations should be given for changes that are rejected. Change The status of the change with a respect to the Change Control Committee: Control Un-reviewed Committee Reviewed Status Follow-Up LOE (hrs) The estimated Level of Effort (LOE) to complete the change request. Migrate Success of migrating code to production: (Pass/Fail) Metric Associated with this checkbox is the Assignee's ID & Date fields. These may be filled automatically with the ID of the current user and the current date when the checkbox is checked or unchecked. Problem Explanation of the problems caused by the change Explanation request. Lessons Explanation of the successful and unsuccessful Learned tactics used during the lifecycle of the change request. Closed By Person who closed the change request. This field may be filled automatically with the current user's ID when the status is changed to "Closed", "Rejected" or "Duplicate". Date Closed Date the change request is no longer being monitored in production. This field may be filled automatically with the current date when the status is changed to "Closed", "Rejected" or "Duplicate".

Detailed Description Paragraph Table (130):
Field Name Field Description Date From The starting date of the date range. If this field is entered, the To Date must be entered. To Date The ending date of the date range. Requester The person who requested the change request Appl. The application area affected by the change request (i.e. Marketing, LIS, Vision) Logged By Portion of the present descriptions who entered the change request into the system. Platform The hardware platform of the system affected by the change request. Source Source of the problem (i.e. Regulatory, ABENDS, Performance, etc.). Function The function affected by the change request. Component What component may the change request affect (i.e. Application Code, Hardware, etc.). Priority The priority of a change request. Category The classification of the change request. Status The status of the change request. A change request can have a status of: New, Assigned, Design, Testing, etc.) Manager Manager of assigned IT Area Assigned To Developer assigned to change request. Both the Assigned 1 and Assigned 2 fields may be queried. IT Area Area assigned to complete the change request (Prod. App. Services, Tech. Services, Client Services, etc.). Target Date Date scheduled to move change into production. Site The site of the change request. Migrate Metric Success of migrating code to production. Prod. Metric Success of code in production.

Detailed Description Paragraph Table (131):
Button Name Button Description IT Area Priority Opens the IT Area by Priority Report. IT Area Status Opens the IT Area by Status Report. Application Priority Opens the Application by Priority Report. Application Status Opens the Application by Status Report. Status by Priority Opens the Status by Priority Report. Manager Priority Opens the Manager by Priority Report. Manager Status Opens the Manager by Status Report. Manager Migration Opens the Manager Migration Report. Manager Opens the Manager Production Report. Production Manager Category Opens the Manager by Category Report. Closed and Opens the Closed and Rejected Report. Rejected Implement Opens the Implementation Report. Recent Chg. Line Opens the Recent Change Line Report. Recent Chg. Detail Opens the Recent Change Detail Reports. Capacity Planning Opens the Capacity Planning Report. Exit Returns the user to the Change Request Log Form.

Detailed Description Paragraph Table (132):
Field Name Field Description CR# Automatically assigned when a new change request is entered. Date Date the change request was entered. This date defaults to the date of the change request entry. Logged By Portion of the present descriptions who entered the change request into the CTT. Requester The person who requested the change request Platform The hardware platform of the system affected by the change request. Appl. The application area affected by the change request. Component What components may the change request affect (i.e. Application Code, Hardware, etc.). Status The status of the change request. A change request can have a status of: New, Assigned, Design, Testing, etc.) Priority The priority of a change request. Category The classification of the change request. Manager Manager of assigned IT Area Assigned To Developer assigned to change request Target Date Date scheduled to move change into production. Actual Date Actual date change is moved into production. Short Description A short (75 chars max) and concise description of the change request. Resolution The resolution to the change request. Developers should include a brief description of the changes made to the code. Explanations should be given for changes that are rejected. Statement of Indicates whether the Statement of Work or Work/Scope Scope Definition has been signed off Definition Checkbox User Acceptance Indicates whether the User Acceptance Testing Testing Checkbox has been signed off. Technical/Code Indicates whether the Technical/Code Review Review Checkbox has occurred. Complete Portion Indicates whether the Complete Portion of the present of the present description has been provided. description Checkbox Complete JCL/ Indicates whether the Complete JCL/DCL and DCL and Programs has been provided. Programs Checkbox Submit Turnover/ Indicates whether the Turnover/Software Software Install/ Install/Panapt Move has been submitted. Panapt Move Distribution List Indicates whether Distribution List Requirements Requirements has been provided. (i.e. TCPIP, Checkbox Special Forms, Microfiche, Electronic Files) Identify Impacted Indicates whether Impacted Systems has Systems been identified. Checkbox Capacity Planning Indicates whether Capacity Planning Checkbox has signed off. Ready to Migrate Indicates whether the change request Checkbox is ready to be migrated to production.

Detailed Description Paragraph Table (134):
Tab: CM PLAN - Proj Project Information for Configuration Management Plan Definition Project information - lists key contacts on the project, Project Configuration Management Board members, and items to be placed under Configuration Management and managed by this plan Purpose Capture project-specific contact information, configuration type information, and Project Configuration Management Board information. It is important to portion of the present description the contacts and responsibilities early in the pro- ject so there are no misunderstandings, and everyone is in agreement on how Configuration Management may be hand- led on the project. Required All Fields Project/ Responsibility: Project Manager Version information Enter project-specific information for this Configuration Management Plan: Platform: the platform that the project may run on. Should match a delivery vehicle, list as "operating system - database system - language". (e.g. VMS - Oracle - C, Fortran) Project: the project/application name. Version: the version of the application. (e.g. 2.0) Production Date: the date that the application is due in production. (e.g. Jan 5, 1998) Configuration Management Plan Owner: the person who may manage and enforce the responsibilities portion of the present descriptioned in the Configuration Management Plan. "Project Responsibility: Project Manager Contacts" Enter names for the project contacts playing the listed roles. These names may automatically populate in subsequent worksheets in the Configuration Management Plan according to the Program Methodology and CMM requirements. If a role does not apply

for your project, enter a space in place of the name. If more than one person plays a primary role, one can enter both names in the field. If a role is not listed, then add it; however, it may not auto-populate into the subsequent worksheets. These contacts should be the people most in- volved in the project, who one would consider "key contacts" involved in the migration process. "To be Primary Responsibility: Functional Lead placed un- der Con- figuration Manage- ment" For each category shown, list the configuration types that may be covered in the Project Configuration Management Plan. The types already listed on the template cover most project needs; verify that they cover the specifics of your particular project. These are the types that may be migrating through the different environments, are subject to review, follow the same approval/migration process, etc.; and may populate on to the subsequent Configuration Management worksheets. The types should cover the "normal" situation (think of the 80% of the 80/20 rule); "exception" situations should be noted separately. Examples for the types: Portion of the present description = design, test, data, support, etc. Database = tables, indices, views, aliases/synonyms, stored procedures, etc. Architecture = application requested architecture extensions Application = code, reports, screens, menus, etc. SCMP tasks: 1.6.1 Identify configuration types "Project Primiary Responsibility: Project Manager Configur- ation Manage- ment Board" Confirm the names of the people who may act as the Project Configuration Management Board (the names automatically populate with contact names listed on the same sheet: Config- uration Management Plan Owner, Development Lead, Test Lead, Tech Lead, Development DBA, Implementation Lead, Operations. The makeup of the CM Board can change, if the project deems necessary. The CM Board is responsible for portion of the present descriptioning the detailed processes on the different tabs and for signing off on the Project Configuration Management Plan developed. They are also responsible for enforcing processes on their teams, and meeting with project management after each major project phase to ensure that changes are completed according to the portion of the present descriptioned Plan. SCMP tasks: 3.6.1 Define control groups; 3.6.2 Approve/disapprove change requests; 3.6.3 Track/implement change request; 4.6.2 Generate/distribute status reports "Meeting Primary Responsibility: Project Manager Dates" and "Meeting Minutes Location" Once the Project Configuration Management Plan has been established and signed off during Design, Project Configura- tion Management Board meetings may be held. They should be held at the end of AT, PT, and Implementation phases - these dates should be taken from the project plan. These meeting dates should be listed in the "Meeting Dates" column. The Project CM Board may meet on the listed dates to review progress made on implementing change requests according to the CM Plan and on action items ensuring CM compliance. The Project Configuration Management Status Agenda ("Status Agenda" tab in the Project Configuration Management Plan) can be used to guide meeting discussions. Minutes from these meetings may be portion of the present descriptioned, and the location of these portion of the present descriptions should be entered in the "Meeting Minutes Location" column. The PS should communicate the meeting dates early in the project to ensure that the meetings may be held on the date noted. References: Project Configuration Management Status Agenda ("Status Agenda" tab in the CM Plan) SCMP tasks: 4.6.1 Maintain records; 4.6.2 Generate/distribute status reports; 4.6.3 Schedule CI reviews; 4.6.4 Perform audits; 5.6.1 Verify security practices Tab: CM* * = Emer, Doc, DB, Arch, Appl Definition Project Configuration Management Plan for the Emer = emergency maintenance (responsibility = Develop- ment) Doc = portion of the present description (responsibility = Development, Test), DB = database objects (responsibility = Tech Support), Arch = architecture extensions (responsibility = Tech Support, Architecture), Appl = application objects (responsibility = Development, Test, Tech, Impl, Operations) created for the project. Purpose Identify the objects to be placed under CM and the stages which they may go through, identify the repository and ver- sion control tool, identify the change request tool that may be used, and portion of the present description the roles/responsi- bilities for migrations to the different environments. It is important to portion of the present description these responsibilities so that it is known who needs to sign off on what tests, who should have authority to write where, and who is responsible for migrations. This may help to ensure that ob- ject migrations are not missed due to misunderstandings. The green WHO columns may automatically populate from the "Project Contacts" listed on the CM PLAN-Proj tab according to the program methodology and CMM requirements. A new project team member should be able to tell exactly what steps to take and who should be involved/notified in order to migrate changes from one environment to the next. Required All Fields SCMP tasks: 2.6.2 Define promotion and migration procedures. "Objects Primary Responsibility: Included" Portion of the present description tab -

Development, Database tab - Tech Support, Architecture tab - Tech Support, Application tab - Development However, Test, Implementation, and Information Delivery may input into each tab, also. This field is pulled from the "Objects Included" field on the CM PLAN-Proj tab. Verify that the list includes all objects that may migrate through the environments, follow the same naming standard, use the same version control tool, follow the same migration procedures, and use the same change re- quest tool. This should be the 80% of the 80/20 rule. The exceptions should be listed in the "Exceptions" filed later in the sheet. Corrections to this field should be made in the "Objects Included" field on the CM PLAN-Proj tab. Any other objects (the 20%) that do not follow the Naming standard, use the Migration/version control tool, or use the Change request tool listed at the top, but do migrate through the development environment should be listed in the "Excep- tions" section of the sheet. The migration path and other information should be filled out for the exceptions, also. SCMP tasks: 1.6.2 Identify project baselines; 1.6.4 Identify configuration units. "Naming Primary Responsibility: (see "Objects Included") Standard" List the location(s) of the naming standard(s) used for the objects listed. The default Alliance Methodology naming standard is listed; any project-specific naming standard should also be listed. This should be the 80% of the 80/20 rule. The exceptions (or 20%) should be listed in the "Exceptions" portion of the description later in the sheet, with that naming standard portion of the present descriptioned. SCMP tasks: 1.6.3 Define naming standards for types "Migra- Primary Responsibility: (see "Objects Included") tion/Ver- sion Con- trol Tool" List the migration and version control tool(s) used. If the tool is only used for one of the functions, indicate that as so. If multiple tools are used to perform these tasks, indicate this, also. Again, this is the 80% of the 80/20 rule. The exceptions (or 20%) should be listed in the "Exceptions" portion of the description later in the sheet, with that migration/version con- trol tool portion of the present descriptioned. SCMP tasks: 2.6.1 Establish platform repositories; 2.6.2 Establish backup/recovery scheme "Change Primary Responsibility: (see "Objects Included") Request Tool" Enter the change request tool used for tracking changes. If multiple tools are used, indicate this and when each tool is used. Again, this is the 80% of the 80/20 rule. The exceptions (or 20%) should be listed in the "Exceptions" portion of the description later in the sheet, with that change request tool portion of the present descriptioned. SCMP tasks: 3.6.1 Define control groups "CM Unit/ The phases listed under the gray heading are the standard test Environ- phases; and the columns to the right apply to the types listed ment" in the "Objects Included" field. Exceptions to that should be listed separately in the "Exceptions" portion of the descrip- tion, with phases listed below and processes listed to the right. These headings can be changed to better fit the project's terminology (e.g. "Unit Test" instead of CT). Each cell to the right of the listed phase has guidelines for the process to be followed on the project. The green WHO fields automatically populate from the "Project Contacts" on the CM Proj tab, according to program methodology and CMM guide- lines. However, these fields may be modified to fit your par- ticular project. Tab: CM Emergency Fixes Emer Primary Responsibility: Development, Tech Support Emergency (EM) fixes are fixes that were discovered in pro- duction, and need to be fixed in production right away. Most of the time, EM fixes go through brief testing (due to time constraints), and are not migrated through all environments. "Production Support" represents the environment where these fixes are made and tested - it is usually separate from the development environment. Prod Sup- Promotion/migration for all objects resulting from an emer- port .fwdarw. gency fix in production: Product- Production Support (EM) to Production tion Production Support (EM) to Component Test Prod Sup- (project should follow migration process defined in "CM port .fwdarw. Appl" from this point) CT (man- ual move) Tab: CM Portion of the present description Doc Responsibility: Development, Test Portion of the present description applies to any/all portion of the present description produced/updated for the project. For example, designs, test conditions and scripts, test data, support procedures, etc. WIP .fwdarw. Promotion/migration for all portion of the present descriptions Final for the project: Work in Progress .fwdarw. Final

Detailed Description Paragraph Table (135):
Tab: CM Database objects, Application objects DB, Appl Responsibility: (Database objects) Tech Support; (Application objects) Development, Test, Operations Database objects include anything related to the storage of data and database objects and functions: tables, views, roles, stored procedures, etc. Application objects include anything developed specifically for the application: screens, windows, programs (online, batch), libraries, etc. *indicates Promotion/migration for database objects

and application base mi- objects to environments: gration level CT .fwdarw. Component Test to Assembly Test AT* AT .fwdarw. Assembly Test to Product Test PT* PT .fwdarw. Product Test to Operational Readiness Test ORT PT .fwdarw. Product Test to Training Training ORT .fwdarw. Operational Readiness Test to Production Produc- tion* Produc- Production to Production Support tion .fwdarw. Prod Support Tab. CM Architecture extensions Arch Architecture extensions are application-specific additions to the existing application architecture that are done by the Tech Support team. This does not include common code developed by the project team - that should be included in the CM Appl tab. Promotion/migration for architecture objects to environments Archi- (owned by Architecture Team) tecture - owned CT .fwdarw. Component Test to Assembly/Product Test AT/PT AT/PT .fwdarw. Assembly/Product Test to Quality Assurance QA AT/PT .fwdarw. Assembly/Product Test to Pilot Pilot AT/PT .fwdarw. Assembly/Product Test to Production (platform development) Production Platform - (owned by platform development teams) owned CT .fwdarw. Component Test to Assembly Test AT AT .fwdarw. Assembly Test to Product Test PT PT .fwdarw. Product Test to Operational Readiness Test ORT PT .fwdarw. Product Test to Training Training ORT .fwdarw. Operational Readiness Test to Production Production Produc- Production to Production Support tion .fwdarw. Prod Support Tab: Sta- Configuration Management Status Meeting Agenda tus Agenda Definition Template agenda to guide discussion during a Project Con- figuration Management status meeting Purpose A Project Configuration Management Status Meeting should be held at the end of Assembly Test, Product Test, and Imple- mentation. The primary goal is to ensure that the Project Con- figuration Management Plan is being followed for all changes, and also to give the project a chance to identify areas for im- provement and act upon them during the project. SCMP tasks: 3.6.1 Define control groups Tab: Com- Configuration Management Compliance Checklist pliance Chk Definition A checklist for projects to use to complete an internal audit on their Configuration Management tasks Purpose Responsibility: all project teams Teams should perform internal audits periodically during the project lifecycle to ensure that processes are being followed, and that Configuration Management tasks have been com- pleted according to SCM Policy. This may also help the team to prepare for external audits. Recommended checkpoints: after Assembly Test, after Product Test. SCMP tasks: 3.6.1 Define control groups Tab: Configuration Management Plan Change Log Change Log Definition Change log/audit trail for the Project Configuration Management Plan itself. Purpose Responsibility: all project teams Provide an audit trail for changes made to the Project Config- uration Management Plan after it has been signed off.

Detailed Description Paragraph Table (149):
Support Center/Operations maintain system baselines approve and implement changes to that baseline Technicnal Support establish the version repositories define packaging and installation procedures assist in migration activities maintain inventory lists Architecture maintain architecture baselines approve and implement changes to that baseline Development migrate components on all platforms maintain inventory lists Test migrate components on all platforms maintain inventory lists Implementation migrate components on all platforms maintain inventory lists Program Management periodically review CM activities and identify CM improvements periodically review individual projects for compliance with program CM process periodically review and recommend improvements to the program CM process Team Leads ensure that CM activities are being performed adhere to CM guidelines Functional Lead move deliverables to the final folder adhere to CM guidelines Project Manager ensure that CM Plans are created for each project

Detailed Description Paragraph Table (155):
Platform Information Platform Type All Description Migration from Component Test to Assembly Test occurs when the Development team successfully completes the Component Test exit criteria. The timing of the migration should be coordinated between all members of the Development project group. If the project involves more than one platform, the cross platform migration should also be coordinated to be sure that units reach the next phase at the appropriate time. For each platform the migration "kit" should include all units required for the project along with any instructional units. The kit should be created and sent to a staging area until approval for installation in the Assembly Test environment is given. Migration to Stage Information Approval to Stage Development Team Member (the approval must be tracked) (1) Exit/Approval CT Exit Criteria Criteria Kit Creation/Trigger If the Kit creation is automated the trigger should come from Performed by (2) the approver to stage as listed above. & (3) If the Kit creation is not automated, then the Development or TS

team should create the kit based upon a portion of the present descriptioned set of procedures. Pre-Migration The pre-migration location for each unit of the migration kit Location should be in an approved library/repository that conforms to the CM repository requirements Post-Migration The post-migration location can be a physically separate Location directory with the appropriate level or security, allowing write access for the kit creation process and read access for the moving of the kits. The post migration location can also be a logical location, where units are tagged with the AT level. Packaging Information Manual/Automated Migration can either be manual or automated. In either case the Package? Tool? process needs to be portion of the present descriptioned and must meet the CM requirements for tracking and recovery. Brief Package Packaging of the CM units should involve a grouping of all Description required units; this grouping should be maintained throughout the entire migration process. This may prevent units from being lost or added during migrations. If multiple units are combined to create a derived product, the creation of the product should be automated by combining like tagged units within the repository (example: a.h, a_sub.pc, a_main.pc, should all be tagged at the CT level). This may prevent the derived product from becoming out of sync with its sub-components in the repository. Only the final product needs to be migrated. Package Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. Verified by The verification should be performed by a Development Project team member, prior to the begirning of Assembly Test, this venfication can be tracked. Internal/External Notification Internal The following teams should be notified upon successful completion of a migration: Development Project Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams. Migration from Stage Information Approval from Stage Development Member (the approval must be tracked) (4) Entrance/ApprovaI AT Entrance Criteria Criteria Kit Move Perf by Moving the kit from the staging environment to the installation (5) area can be performed by any person from one of the following teams: Development, Operations, or TS. Movers need to be certain that the appropriate approval has been given prior to moving the kit. Kit Install Perf by Installation of the kit into the new environment can also be (6) performed by multiple groups. Consideration sbould be given to the level of system security access required to perform the installation. Whenever a significant level of access is required, the installation process should be limited to either the TS team or Operations. Pre-Migration The pre-migration location should match the post migration Location location listed above for the Migration to Stage Post-Migration The post-migration location should be a physically separate Location environment from the CT environment whenever feasible and cost effective. This location should mirror the production environment as closely as possible. Un-Packaging/Installation Information Manual/Automated For complex systems and installations requiring a significant Package? Tool? level of access the process should be automated. Manual processes may require explicit directions and a more rigorous verification process. Brief Package Whether the installation process is manual or automated, the Description process should be clearly portion of the present descriptioned. All units should have a specific location on the destination server. Migration from Stage Information The installation process should take into account factors such as space, currently running executables, overwriting existing units, and ?? Install Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. For manual process the verification should be more extensive Verified by The verification should be performed by a Development Project team member, prior to the beginning of Assembly Test. This verification can be tracked. Internal/External Notification Internal The following teams should be notified upon successful completion of a migration: Development Project Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams.

Detailed Description Paragraph Table (156):
Platform Information Platform Type All Description Migration frorn Assembly Test to Product Test occurs when the Development team has successfully completed the Assembly Test exit criteria. The timing of the migration should be coordinated between Development and Test. If the project involves more than one platform, the cross platform migration should also be coordinated to be sure that units reach the next phase at the appropriate time. For each platform the migration "kit" should include

all units required for the project along with any instructional units. The kit should be created and sent to a staging area until approval for installation in the Product Test environment is given. Migration to Stage Information Approval to Stage Development Team Member (this approval must be tracked) (1) Exit/Approval AT Exit Criteria Criteria Kit Creation/Trigger If the Kit creation is automated the trigger should come from Performed by (2) the approver to stage as listed above. & (3) If the Kit creation is not automated, then the Development or TS team should created the kit based upon a portion of the present descriptioned set of procedures. Pre-Migration The pre-migration location can be a physically separate Location directory with the appropriate level of security or it can be a logical environment in which the units are tagged with the appropriate migration level. Post-Migration The staging environment can be a physically separate directory Location with the appropriate level of security or it can be a logically separate environment in which the units are tagged with the appropriate migration level. Packaging Information Manual/Automated Migration can either be manual or automated. In either case the Package? Tool? process needs to be portion of the present descriptioned and must meet the CM requirements for tracking and recovery. Brief Package Packaging of the CM units should involve utilizing the same Description grouping as the migration from CT to AT, this may prevent the introduction of new units or the loss of required units. Migration to Stage Information If multiple units are combined to create a derived product then only the derived product needs to be migrated. Some environments may require the product to be created differently for each destination environment, in this case the sub-components need to be migrated as well. Package Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. This procedure may require an extra step during the actual packaging to create an audit log identifying the status of the migration. Verified by The verification should be performed by a Development Project team member, prior to notifying Test. Internal/External Notification Internal The following teams should be notified upon successful completion of a migration: Test Team External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams. Migration from Stage Information Approval from Stage Development Member (this approval must to be tracked) (4) Entrance/Approval PT Entrance Criteria Criteria Kit Move Perf by Moving the kit ftom the staging environment to the installation (5) area can be performed by any person from one of the following teams: Development, Operations, or TS. Movers need to be certain that the appropriate approval has been given prior to moving the kit. Kit Install Perf by Installation of the kit into the new environment can also be (6) performed by multiple groups. Consideration should be given to the level of system security access required to perform the installation. Whenever a significant level of access is required, the installation process should be limited to either the TS team or Operations. Pre-Migration The pre-migration location should match the post migration Location location listed above for the Migration to Stage Post-Migration The post-migration location should be a physically separate Location environment from the CT environment whenever feasible and cost effective. This location should mirror the production environment as closely as possible. Un-Packaging/Installation Information Manual/Automated For complex systems and installations requiring a significant Package? Tool? level of access the process should be automated. Manual process may require explicit directions and a more rigorous verification process. Brief Package Whether the installation process is manual or automated, the Description process should be clearly portion of the present descriptioned. All units should have a specific location on the destination server. The installation process should take into account factors such as space, currently running executables, overwriting existing units, and?? Install Verification Verification Check A simple procedure should be defined to allow for verification of a successful migration. This procedure may require an extra step during the actual packaging to create an audit log identifying the status of the migration. Migration from Stage Information For manual processes the verification should be more extensive Verified by The verification should be performed by an Test member, prior to the beginning of Product Test. Internal/External Notification Internal The following teams should be notified upon successful completion of the migration: Test External At this stage no outside communication is required except for project status purposes. This task should be incorporated into the project status meeting in order to notify business partners and other project teams.

Current US Cross Reference Classification (2):

707/100

WEST

Generate Collection | Print

L10: Entry 1 of 2 File: USPT Jul 8, 2003

DOCUMENT-IDENTIFIER: US 6591272 B1
TITLE: Method and apparatus to make and transmit objects from a database on a server computer to a client computer

Abstract Text (1):
Contents of databases are translated into objects by reading the database schema metadata to determine data interrelationships and create objects with nominal human to computer interaction. Metadata for any number of databases is normalized in a standardized view. Skeleton code templates representative of final classes to be produced are accessed and merged with the standardized view. Source code for the class of the objects is then generated. At runtime, data objects are then produced by encapsulating the metadata and data values. Communication between database instances and a client computer consists of metadata and database row values., Rows from database tables and the corresponding metadata are transmitted from the server to the client computer in one logical network operation. The final distributed objects are then assembled into the optimal format required by the client computer. To update, delete or create new persistent objects, the reverse process occurs.

Brief Summary Text (14):
In the prior art, manual creation or manipulation of SQL is also typically needed to populate the objects from the database though this intermediate object-relational mapping layer. This SQL can be specific to the brand or vendor of the database, making migration from one database type to that of another vendor costly and problematic. In the prior art, the combination of numerous, heterogeneous databases from different database vendors was either time-consuming, error-prone, problematic, inefficient, or not possible.

Brief Summary Text (18):
In the present invention, an automated, expert method, system and program product that translates and transmits metadata and data from database tables into familiar and customary objects desired is disclosed. The method comprises the steps of reading the definitional elements of the databases to determine data types and interrelationships between relational data elements. These data interrelationships and data types are assembled in a vendor-neutral standardized view of the database schemas and the plurality of all the possible logical objects contained therein in the databases are created. Template definitions generically represent the classes of the objects desired.

Brief Summary Text (20):
Source code for the classes is then generated from the standardized view when merged with the prepared template definitions. The source code is then compiled into binary executable form into the classes desired. Pseudo-objects are then produced by dynamic generation and execution of pre-optimized SQL, enveloping values that result from execution of the generated prepared SQL statements. Result sets from said associated prepared statement operations from the appropriate database tables and rows are normalized into a standard format, then combined with metadata from the database schemas. The pseudo-objects are then ready for transmission to the client computer or the requester of the objects desired. The present invention also relates to a method of communicating elements of a database table between a server computer and a client computer. A pseudo-object is generated by the server computer with the pseudo-object comprising rows from singular database tables, or optimized joins between multiple related database tables, that comprise the object desired. The plurality of datatypes present in the relational databases are normalized into a singular standardized form

to prepare the data for transmission to the requestor of the object. Metadata of the elements where the metadata is the relationship between the data elements is also generated by the server computer. The metadata and normalized pseudo-object data are transmitted from the server computer to the client computer in a single logical transmission.. At the client computer, the elements are assembled into the final objects from the pseudo-objects and metadata received, into the format required by the software on the client computer without runtime overhead on the database or middle-tier server computers.

Detailed Description Text (7):
The foregoing method can be analogized to the following in the word processing area. Assume the documents 20(a-e) have been created by various different application programs such as Word, WordPerfect, PageMaker, Claris, etc. The standardized view of each particular document 24, may be the DOS text version of those characters without the specific attribute codes or metadata produced by the respective programs but with the text indicating where the attribute codes should be placed. For example, different word processing programs generate different code for the attribute of "bold" or "underscore". The standardized view of the document 24 simply has the reference to the words that constitute the document as well as to indicate that that particular word or phase is to be "bolded". The code 26 that is so generated would then provide the specific bold code for that object or that version of the document desired.

Detailed Description Text (8):
It is the use of templates 22 which are not object-specific combined with standardized view of object specific tables 24 to convert into code 26 (a-z), such that after compiling and execution pseudo-objects 30 of the classes desired and its associated metadata 31, is produced at run time, that is the basis of the first method of the present invention. The software 21 to perform the foregoing method is set forth in the software modules entitled: OSFMain.java and OSFGenerate.java, within the Principles of Operation set forth on the CD-ROM filed herewith.

Detailed Description Text (9):
Referring to FIG. 3, once the code 26 (on FIG. 2) is compiled and loaded upon server computer 12, object access or update requests can originate from client computer(s) 16 over network 14. Server computer 12, in response to an object access request, generates the pseudo-object 30 and the associated metadata 31, in a particular object of choice is generated. During execution, subsets of database tables in the databases 20(a-e) (on FIG. 2) are enveloped by code 26 (on FIG. 2) to become a pseudo-object 30 desired, along with its associated metadata 31, and is transported as a single logical network packet unit transmitted synchronously or asynchronously over the network 14 to the client computer 16. The pseudo-object 30 so transmitted over the network 14 with metadata 31 is received as the received pseudo-object 32 and received metadata 33 at the client computer 16. However, as can be seen from the previous description, the received pseudo-object 32 is in essence data a subset of rows from database tables in databases 20(a-e) of the relevant entries comprising a plurality of entries. Thus, the received pseudo-object 32 at the client computer 16 is then assembled into a plurality of true objects 34(a-f) in an object, or block of objects; or hierarchical or "tree" object representation 38 having the specific relationships between the objects 34(a-f). The object(s) 38 then can be presented to the user through conventional display means such as HTML etc.

Detailed Description Text (10):
The manner by which the pseudo-object 30 and metadata 31 is transmitted over the network 14 in which the data and their relationship is transmitted in a single packet unit is as follows. The pseudo-object 30, is as previously described, comprises a plurality of the values 34(a-f) of the object 38, in the user interface of the client computer 16. The relationship between these objects 34(a-f), called metadata, was transmitted along with the pseudo-object 30. In the logical network transmission pseudo-object 30 to the client computer 16, the pseudo-object 30 comprised data values of the objects 34(a-f). In addition, the metadata 31 indicating the relationship between the pseudo-objects 34(a-f) was also transmitted.

Detailed Description Text (11):
At the client computer 16, the received pseudo-object 32 is assembled to retrieve the data values 34(a-f), and the metadata is then used to place these data values 34(a-f)

into the user interface of the client computer 16, or in the case of creation of new objects, from the client computer 16 to the server computers 12 containing the database tables in databases 20(a-e) (of FIG. 2). In this manner, an efficient means of transmitting a number of database elements from a server computer 12 to a client computer 16 and new objects or object updates or object deletions from client computer 16 to server computer 12 is accomplished.

Detailed Description Text (12):
Thus, in the present invention, where the client computer 16 makes a single request and a single pseudo-object 30 with metadata is transmitted over the network 14 with the received pseudo-object 32 thereafter assembled and placed into the user interface of the client computer 16, only two uses of the network 14 are made. This greatly reduces traffic on the network 14 and dramatically reduces CPU processing time requirements on server computer 12. Conversely, when new objects are created, a single object 34 is created on client computer 16, is transmitted over network 14 to server computer 12 where the pseudo-object 30 is created by metadata 31 on server computer 12 to create appropriate new table rows in database tables in databases 20(a-e) with only two uses of network 14. Further, the client can assemble the objects into the precise format desired by the user interface object being populated by object data; such as tree or hierarchical format, or block or grid data format. The software to perform the foregoing method is set forth in the software modules entitled: OSFORBStream java, OSFORBStreamObjectjava, and OSFORBStreamExceptionjava which are on the CD-ROM filed herewith.

Detailed Description Text (40):
The easiest way to define an OSFORBStream is to provide a functional summary of the four sub-types of OSFORBStreams. Server side, transmit OSFORBStreams are used to package the raw records read from the flat database tables. Metadata is sent along with the raw table records to identify the base table records, identify the primary and foreign keys and data fields and the relationships between key fields between the table records.

Detailed Description Text (41):
The reason metadata is sent is so client-side objects do not need to have server-side definition files imported into the client-end application. This makes OSFORBStreams dynamic so clients or object requesters can be insulated from changes to the backend database tables. A very useful feature but was complex to implement in practice. Client/requestor side, receive OSFORBStreams process the server side, transmit OSFORBStreams. The aforementioned metadata is isolated, verified then used to assemble the objects into the form desired by the requestor. Client/requestor side, transmit OSFORBStreams are used to create, delete and update server-side objects. Server side, receive OSFORBStreams are used to disassemble the objects back into the corresponding flat database tables so the appropriate object delete, object insertion or object attribute update operations can be performed.

Detailed Description Text (60):
After a transmit-type OSFORBStream is instantiated, metadata needed for object assembly is appended to the OSFORBStream. Then result objects from the individual table reads are stringified, appended to the OSFORBStream and transmitted to the client or consumer of the object (as in the case of server middleware).

Detailed Description Text (66):
Metadata is sent at the front of the OSFORBStream so rows can be merged and redundant fields eliminated during object assembly (all foreign keys are by definition redundant and must be removed from the child rows during object assembly). This metadata takes the form of OSF KeyMaps in external string form and are part of the OSFORBStream header. Static helper methods in the OSFORBStream class assist with the construction of these OSFORBStream headers.

Detailed Description Text (222):
The idea was to simplify testing and expedite release planning and migration to production of OSF-based solutions, an arduous process in some large, enterprise environments, which can be painfully bureaucratic.

Detailed Description Text (411):

When the parameters outlined above have been entered, click the far-left toolbar icon. The parameters will be read from the above panels and a connect to the target database will proceed.

Detailed Description Text (643):
Since ObjectServerFactory knows the datatype, precision and scale of each column, it writes into the TestDepartmentEmployeePersistence class a list of arbitrarily high values that are not likely to exist in the source database:

Detailed Description Text (702):
Finally, the most complex are the object block streams which may contain a single object or multiple objects and because these streams contain metadata about the objects used for final assembly of the objects in the format desired by the requestor.

Detailed Description Paragraph Table (111):
Target in Skeleton Function and Operation by Template file OSFGenerate ##Package## -> package target (0) ##TableObjectName## -> normalised table name (1) ##TABLENAME## -> insert table name in UPPER CASE (2) ##COLUMNNAMES## -> insert all column names in UPPER CASE (3) ##KEYFIELDSAND- -> array of ints defining which cols are SORTORDER keys (4) ##tableobjectname## -> all lower case normalised table name (5) ##ObjectName## -> upper and lower case normalised or specified object name (6) ##objectname## -> lower case normalised or specified object name (7) ##BaseTableObjects## -> enumerate all base table objects (8) ##inheritanceblock## -> recursively invoke parseSkeletonRecord() until ##endinheritanceblock## is encountered in the input template stream (9) ##index## -> insert an index counter, scoped within a given ##codeblock## (10) ##AttributeName## -> attribute name as a java-style class- - first byte upper case (11) ##attributeName## -> attribute name as a java-style method-- first byte lower case (12) ##ATTRIBUTENAME## -> UPPER CASE attribute name (13) ##attributeblock## -> recursively invoke parseSkeletonRecord() until ##endattributeblock## is encountered in the input template stream (14) ##attributeonlyblock## -> same as an ##attributeblock## but with no key fields (15) ##allkeyattributeblock# -> same as an ##attributeblock## but # with only key fields (16) ##keyFields## -> insert key fields as java-style method-- first byte lower case (17) ##MAXKEYCOUNT## -> insert nonnegative numeric integer constant of all object keys (18) ##ATTRIBUTECOUNT## -> insert nnic of count of attributes of object, including keys (19) ##parentKeyFields## -> insert key fields of top-level table object ONLY-- first byte lower case (20) ##attributesNoKeys## -> insert attribute names only, no primary or secondary keyfields (21) ##attributeNamesKeysQua -> all attributes, but at the end of a lfied## key field append keysuffix_ (22) ##keymap## -> insert metadata about key fields of underlying base tables (23) ##OBJECTNAME## -> UPPERCASE normalised or specified object name (24) ##counter+init## -> special tag to initialise a special internal counter. No output. (25) ##counter## -> insert the current value of the above counter, then increment (26) ##registryentrycount## -> insert the count of registry entries written (27) ##allcolumnblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (28) ##COLUMNNAME## -> recursively insert a singular column name in UPPER CASE (29) ##TABLE## -> recursively insert a singular table name in UPPER CASE (30) ##entrycount++## -> increment registry entry count-- no output (31) ##allattributeblock## -> recursively invoke parseSkeletonRecord() until ##endcolumnblock## is encountered in the input template stream (32) ##DEFAULTMIN## -> based on datatype and attribute length, insert a reasonable default minimum value (33) ##DEFAULTMAX## -> based on datatype and attribute length, insert a reasonable default minimum value (34) ##VALIDATIONTYPE## -> based on datatype insert the validation type as defined in the OSFRulesObject base class (35) ##fieldlength## -> insert the maximum field length (36) ##picklistcandidates## -> insert picklist candidates from table scan or default string (37) ##iso639language## -> insert the current two byte iso639 language string (38) ##LANGUAGE## -> insert the current language descriptor (39) ##AttributeNameExpanded -> add a space before the 2nd through n capitals in an attribute name and then insert (40) ##language## -> insert the current language descriptor, in lower case (41) ##picklistvalues## -> insert all picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (42) ##picklistvalue## -> insert a unique picklist value guraranteed to be unique (43) ##picklistvalues## -> insert all unique picklist values (multiple lines) or if no picklist exists for this column, suppress output of the record (44)

##databaseblock## -> recursively invoke parseSkeletonRecord() until ##enddatabaseblock## is encountered in the input template stream, setting currentdatabase_ on each interation for each instance on the OSFDatabase list (45) ##DBLOGICALNAME## -> insert in upper case the intenral logical name of the currentdatabase_ (46) ##DBOWNER## -> insert in upper case the ownername of the currentdatabase_ and continue with further replacements (47) ##DBPASSWORD## -> insert in the case entered the password of the owner in the currentdatabase_ object and continue on with further replacements (48) ##DBTYPENAME## -> insert in upper case the jdbtools type name of the currentdatabase_, carry on with further replacements (49) ##DBSERVER## -> insert in the case entered the hostname or IP address in the currentdatabase_, carry on with further replacements (50) ##DBPORT## -> insert IP connect port in the currentdatabase_ object, carry on with further replacements (51) ##DBINSTANCE## -> insert in the case entered the instance name or SID in the currentdatabase_ object, carry on with further replacements (52) ##DBOWNER## -> insert in the case entered by the user the owner / user name in the currentdatabase_ object, carry on with further replacements (53) ##MINKEYCOUNT## -> insert count of keys for a partially qualified read = key count of top level parent (54) ##hasparentconstraint## -> table is part of a relation / has a parent or owning table (55) ##testvalues## -> based upon current object context, insert a list of test attribute values (56) */ ##attributename## -> attribute name as an automatic declaration (57) ##attributenamekeysqual -> all attributes, lower case, at the ified## end of a key field append a lower case keysuffix_ (58) ##javadatatype## -> insert an appropriate Java data type depending on the normalised internal datatype (59) ##initializer## -> insert an appropriate initialiser depending on the normalised internal datatype (60) ##JavaPrimitiveObject## -> insert an name suitable for use in conversion methods (61) ##INTERNALDATA- -> insert the internal datatypes based TYPES## on the current _table (62)

Current US Cross Reference Classification (1):
707/100

CLAIMS:

3. A method of communicating elements of a database, having a metadata, between a server computer and a client computer, comprising: generating a pseudo-object by said server computer, said pseudo-object comprising data of said elements; generating metadata of said elements, wherein said metadata is relationship of said data of said elements, wherein said generating step further comprising: reading said metadata of said database; translating a list of said metadata to a standardized view of said database; accessing skeleton code templates representative of final classes to be produced; generating source code for the classes of the objects desired, and scripts to compile said classes into executable form; producing said object desired by enveloping said data and metadata; transmitting said pseudo-object and metadata from said server computer to said client computer; and assembling said elements to final distributed objects by said client computer from said pseudo-object and metadata received.

6. An article of manufacture comprising: a computer usable medium having computer readable program code embodied therein configured to translate metadata of a database into objects desired, the computer readable program code in said article of manufacture comprising: computer readable program code configured to cause a computer to read said metadata of said database to determine characteristics of said database and their relationship; wherein said computer readable program code further comprises; computer readable program code configured to generate a pick list based upon an inversion of elements of said database; and computer readable program code configured to generate a script containing all unique foreign language strings to be translated; computer readable program code configures to cause a computer to assemble a list of the metadata of said database and their relationship in a pseudo-standardized view of said database; computer readable program code configured to cause a computer to read skeleton code templates representative of final classes to be produced; computer readable program code configured to cause a computer to generate source code for the class of the object desired; and computer readable program code configured to cause a computer to produce said objects desired by enveloping values assembled in said templates.

7. An article of manufacture comprising: a computer usable medium having computer readable program code embodied therein configured to communicate elements of a database table between a server computer and a client computer, the computer readable program code in said article of manufacture comprising: computer readable program code configured to generate a pseudo-object by said server computer, said pseudo-object comprising data of said elements and to generate metadata of said elements, wherein said metadata is relationship of said data, wherein said computer readable program code configured to generate an object by said server computer further comprising: a computer usable medium having computer readable program code embodied therein configured to translate elements of a relational database table into objects desired, the computer readable program code in said article of manufacture comprising: computer readable program code configured to cause a computer to read said elements of said database table to determine values of said elements and their relationship; computer readable program code configured to cause a computer to assemble a list of the values of said elements and their relationship to a standardized view of said database table; computer readable program code configured to cause a computer to access skeleton code templates representative of final objects to be produced; computer readable program code configured to cause a computer to generate source code for the class of the object desired; and computer readable program code configured to cause a computer to produce said objects desired by enveloping values assembled in one of said templates; computer readable program code configured to transmit said pseudo-object and said metadata from said server computer to said client computer; and computer readable program code configured to assemble said elements by said client computer from said object and metadata received.

WEST

End of Result Set

Generate Collection | Print

L11: Entry 1 of 1 File: USPT Feb 18, 2003

DOCUMENT-IDENTIFIER: US 6523027 B1
TITLE: Interfacing servers in a Java based e-commerce architecture

Detailed Description Text (277):
FIG. 15 illustrates a method 1500 for handling events in a system. In operation 1502, an event which includes metadata is recognized. Next, in operation 1504, the metadata of the event is read and, in operation 1506 a table look-up is performed for information relating to the event based on the metadata. The information includes a severity of the event and further information such as a type of the event, and a location where the event occurred. In operation 1508, a message is displayed either in-line in a currently depicted display or in a separate display based on the severity of the event.

Detailed Description Text (1389):
Lightweight Directory Access Protocol (LDAP) is the underlying protocol used by Site Server Membership to communicate with the Membership Directory. LDAP was designed to be the standard Internet protocol for accessing directory services. LDAP runs on TCP/LP networks and is independent of platform, allowing directory-based information to be shared across operating systems. Site Server Membership implements an LDAP service for reading and writing information to the Membership Directory database.

Detailed Description Text (3529):
FIG. 132 depicts a method 13200 for initializing a database used with an issue tracker. The issue tracker receives information relating to a plurality of issues from a plurality of users, displays the information relating to the issues, and allows the browsing of the information relating to each of the issues. To initialize the database, the information relating to the issues is stored in a first database in operation 13202. A second database is provided in operation 13204. The second database stores tables including a plurality of user interfaces and/or application logic for accessing the information in the first database. The tables of the second database are reconfigured in operation 13206 upon migrating the first database from a first folder to a second folder.

Detailed Description Text (3530):
As an option, a copy of the tables may be stored after being reconfigured. As another option, changing of a title of the first database may also be allowed upon migration from the first folder to the second folder. Additionally, the information relating to the issues may also be allowed to be edited.

Detailed Description Text (3534):
The Issue Tracker tool is comprised of two Microsoft Access databases. ReTA Issues DB--Client.mdb, which provides the user interfaces and supporting application logic and ReTA Issues DB.mdb, which contains the actual Issue Tracking data. To access the issue data the client database contains linked Access tables that actually reside in the second database. In order to function correctly these tables must be reconfigured so that the location references are correct. This step needs to be done every time the databases are moved to a difference file system folder.

Current US Cross Reference Classification (2):
707/100

WEST

Generate Collection | Print

L15: Entry 2 of 4 File: USPT Nov 27, 2001

DOCUMENT-IDENTIFIER: US 6324543 B1
TITLE: Dynamic object migration method using proxy object links to support automatic object distribution in an object-oriented environment

Detailed Description Text (47):
The migration method on the local proxy serializes the object, its proxies and the proxies of any objects which it calls and sends it to the remote node for execution. The JVM on the remote machine calls the "ReadObject" method to read the serialized objects, and registers the remote proxies with the Remote Method Invocation (RMI) Registry. Techniques for reading and writing serialized objects across a network via a TCP/IP socket and for interacting with the RMI Registry are well-known. In addition, the remote proxy (generated in step 6) for the migrated object is sent to the remote node.

Current US Original Classification (1):
707/200

WEST

Generate Collection | Print

L15: Entry 3 of 4 File: USPT Jul 31, 2001

DOCUMENT-IDENTIFIER: US 6269382 B1
**** See image for Certificate of Correction ****
TITLE: Systems and methods for migration and recall of data from local and remote storage

Detailed Description Text (48):
Referring now to FIG. 6, a top level block diagram illustrating the processing of I/O requests involving files with remotely stored attributes is illustrated. This figure has particular applicability to processing I/O requests for files in the migrated state (e.g., they have at least one attribute stored only on remote storage). Processing of files in the pre-migrated state is discussed below. In the context of this invention, an I/O request is any operation that may be performed by an I/O system that implements the present invention. Thus, the definition of I/O request goes far beyond the mere reading data from and writing to files. In some situations, an I/O request may trigger other actions not associated with traditional I/O operations, such as calling a phone number when a particular file is accessed. Within the context of this invention, the term is intended to be interpreted broadly

Current US Original Classification (1):
707/204

WEST

End of Result Set

☐ Generate Collection | Print

L25: Entry 2 of 2 File: USPT Oct 12, 1999

DOCUMENT-IDENTIFIER: US 5966717 A
TITLE: Methods for importing data between database management programs

Drawing Description Text (3):
FIG. 1B illustrates a map field window, representing the window through which the user may specify a custom mapping between an import item and a target data field of the target database.

Drawing Description Text (7):
FIG. 4 illustrates, in accordance with a preferred embodiment of the present invention, the steps involved in custom mapping an import item.

Detailed Description Text (9):
Note that the user is able to employ the same target window for custom mapping, which is unlike certain prior art techniques wherein the user is required to call up another map field window, e.g., map field window 150 of FIG. 1B, for custom mapping of an import item. Additionally, the action required to perform custom mapping is very similar to the action taken by the user in entering data, i.e., simply placing the data item in the appropriate target data field. Advantageously, the user is able to perform data import with less training and a greater level of efficiency since not only does the import screen look familiar to the manual entry screen with which the user is accustomed, but the actions required for custom mapping is also analogous to actions required for manual data entry, which are typically already familiar to the user.

Detailed Description Text (20):
Irrespective of the format of the target window employed for data import, it is highly preferable that the target window be arranged to maximize leverage of the user's familiarity with the database management program, i.e., to permit the user to perform the import process using a screen layout with which the user is familiar and without requiring the user to learn a whole new import screen. For usability reasons, it is also highly preferable, as mentioned earlier in connection with FIGS. 2A and 2B, that custom mapping and unmapping be performed with this target window without requiring the user to resort to yet another window, as in the case of prior art FIG. 1B.

Detailed Description Text (21):
In step 306, the import items are mapped into the corresponding target headers/target data fields of the target screen. As mentioned earlier, some of the import items may already have been mapped by default. Import items which require custom mapping, whether already mapped into the target window or still unmapped, are then mapped using the displayed target window and the list of import items displayed in the import window. Preferably, the mapping is performed in a visually intuitive manner such as dragging and dropping, or highlighting an item and clicking on its destination to perform the move.

Detailed Description Text (22):
Note that the invention does not require the user to focus on one piece of import data at a time. In the prior art, the user must select an import item in the prior art import window (e.g., import window 102 in the example of FIGS. 1A and 1B), switch to a different screen to perform custom matching (e.g., via the "MAP TO" mechanism), and return back to the prior art import window to perform mapping of other import items.

The present invention permits the user to perform custom mapping on multiple data fields in the same import window without switching from one window to another window and back, thereby improving efficiency and ease of use.

Current US Original Classification (1):
707/204